Soluções Neurais de Equações Algébricas de Riccati

Annabell D.R. Tamariz e Celso P. Bottura

***Abstract*—Neste artigo resolvemos neuralmente as equações algébricas de Riccati discreta (*EARD*) e contínua (*EARC*) no tempo. Para o caso discreto, nos baseamos em nossa proposta de abordagem que obtem os modelos dinâmicos neurais que descrevem a *EARD* utilizando uma Rede Neural Recorrente multicamada (*RNR*). Para o caso contínuo, apoiados nos modelos dinâmicos neurais que descrevem a *EARC*, fazemos uma implementação computacional que além de calibrar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Apresentamos vários exemplos de aplicação ao problema de projeto de regulador linear quadrático.**

***Index Terms*—Sistemas Lineares, Redes Neurais, Equação de Riccati, Controle linear quadrático, Espaço de Estado.**

I. Introduction

O termo genérico **"Equação de Riccati"** pode significar qualquer classe de matrizes: quadrática, algébrica ou diferencial ou a diferenças finitas de tipo simétrico ou não simétrico, surgida no estudo de sistemas dinâmicos contínuos ou discretos no tempo [4]. As equações de Riccati, surgem naturalmente numa ampla variedade de situações e têm grande utilidade na análise e projeto de sistemas de controle.

Ambas equações (discretas e contínuas no tempo) desempenham um papel fundamental na solução de problemas de Controle Linear Quadrático Gaussiano, Filtro de Kalman, modelagem no Espaço de Estado de Séries Temporais e de Sistemas e em muitos outros ramos da matemática, estatística, engenharia e economia.

Uma Rede Neural Artificial (*RNA*) é uma estrutura de processamento de informação distribuída paralelamente na forma de um grafo direcionado, com algumas restrições e definições próprias, consistindo de neurônios com interconexões sinápticas e enlaces de ativação, vide [9].

Na literatura de controle, dá-se muita atenção aos problemas relacionados com o projeto do regulador linear quadrático (RLQ). O problema de projetar um sistema de controle linear realimentado, minimizando um índice de desempenho quadrático, pode, por exemplo ser reduzido ao problema de obter uma solução definida não negativa da equação algébrica de Riccati. Apesar de existirem algoritmos paralelos que calculam a solução de Riccati muito mais rápidamente que os algoritmos seqüenciais, e de existirem muitas referências a pesquisas para resolver sistemas de equações lineares e problemas relacionados com *RNA*, [5] [7] [8], referências tratando da solução neural da equação matricial de Riccati discreta são escassas.

Neste artigo resolvemos neuralmente as equações algébricas de Riccati discreta (*EARD*) e contínua (*EARC*) no tempo. Para o caso discreto, nos baseamos em nossa proposta de abordagem que obtem os modelos dinâmicos neurais que descrevem a *EARD* utilizando uma *RNR*, [2]. Para o caso contínuo, apoiados nos modelos dinâmicos neurais que descrevem a *EARC*, [11], fazemos uma implementação computacional que além de calibrar nossa solução neural para este caso, permite validar a proposta de abordagem discreta que fizemos e também implementamos. Apresentamos vários exemplos de aplicação ao problema de projeto de regulador linear quadrático.

Nosso objetivo neste trabalho é resolver problemas de controle ótimo *RLQ* para sistemas lineares discretos e contínuos no tempo utilizando *RNR*. Temos em mente o objetivo de criar condições para o controle em tempo real de sistemas com a intenção de que as soluções neurais já desenvolvidas e aqui implementadas por software sejam posteriormente implementadas em hardware microeletrônico.

II. Equação de Riccati Discreta no Tempo

Para motivação consideremos o seguinte sistema controlável linear invariante e discreto no tempo, definido pela equação

$$x_{k+1} = Ax_k + Bu_k, \tag{1}$$

onde $x_k \in \Re^n$ é o vetor de estado do sistema e $u_k \in \Re^m$ é o vetor de entrada de controle, $A \in \Re^{n\times n}$ e $B \in \Re^{m\times m}$ são matrizes constantes conhecidas de dimensões apropriadas associadas com x_k e u_k respectivamente. A função de custo quadrática associada com este sistema pode ser definida como,

$$J = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k), \tag{2}$$

onde $Q > 0$ e $R > 0$ são matrizes de ponderação conhecidas, simétricas e definidas positivas para x_k e u_k respectivamente; esta função deve ser minimizada com a restrição (1). Um controlador ou lei de controle com realimentarão de estado linear, $u_k = -Kx_k$ resultante em [12], pode ser aplicado na equação (1) e obtemos o sistema em malha fechada com a seguinte expressão:

$$x_{k+1} = (A - BK)x_k, \qquad x_0 \tag{3}$$

onde a matriz de ganho realimentada ótima pode ser calculada como:

$$K = (B^T P B + R)^{-1} B^T P A \tag{4}$$

Núcleo de Pesquisa e Desenvolvimento em Informática, Universidade Cândido Mendes, Campos dos Goytacazes, RJ 28040-320.
Telf: +55 22 2733–4100, e-mail: annabell@ucam-campos.br
Laboratório de Controle e Sistemas Inteligentes, Universidade Estadual de Campinas, Campinas, SP ???.
Telf: +55 19 3788–3852, fax: +36 1 463–4112, e-mail: bottura@dmcsi.fee.unicamp.br

e P é uma matriz simétrica, definida não negativa que pode ser obtida através da solução da $EARD$:

$$A^T P(I+SP)^{-1}A - P + Q = 0\ , \tag{5}$$

onde A, Q e S são matrizes reais quadradas com Q e S simétricas e A não singular, I corresponde à matriz identidade. A equação (5) corresponde à forma equivalente da $EARD$. A matriz S, para a equação (5), é definida como:

$$S = BR^{-1}B^T \tag{6}$$

e R é uma matriz não singular,[1] [10]

III. Equação Dinâmica Neural de Riccati Discreta

Nesta seção, vamos apresentar nossa propostas para resolver a $EARD$ definida em (5), usando RNR. Posteriormente vamos apresentar os respectivos resultados numéricos, do problema Neuro-Riccati proposto, obtidos com nossa implementação computacional neural.

Nosso problema neural sería definido com a seguinte função objetivo:

$$\begin{array}{rcl} \min E(P,L) & = & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}[G(P)] + e_{ij}[H(P,L)] \\ & onde & \\ G(P) & = & A^T P(I+SP)^{-1}A - P + Q = 0 \\ H(P,L) & = & LL^T - P = 0 \end{array} \tag{7}$$

Nosso objetivo será, com estas duas equações inclusas na função objetivo, obter as equações dinâmicas neurais para resolver, o problema de controle ótimo apresentado. As equações dinâmicas neurais para resolver este tipo de problemas em forma geral são descritas por, [5]:

$$\frac{dP(t)}{dt} = -\eta_p \frac{\partial E}{\partial P} = -\eta_p W_1 \qquad P(0) = P^T(0) \tag{8}$$

$$\frac{dL(t)}{dt} = -\eta_l \frac{\partial E}{\partial L} = -\eta_l W_2 \qquad W_2(0) \neq 0 \tag{9}$$

onde a derivada de uma função de valor escalar E em relação a uma matriz é definida por

$$\frac{\partial E}{\partial P} = \left[\frac{\partial E}{\partial p_{ij}}\right]_{n\times n} \qquad i,j = 1,...,n$$

Na definição das equações em (8), $P(t)$ e $L(t)$ são matrizes de ativação de estado da RNR, $\eta_p, \eta_l > 0$ são as taxas de aprendizado e $W_1 = [w_{1,ij}]_{n\times n}$ e $W_2 = [w_{2,ij}]_{n\times n}$ são definidas para todo $i,j = \overline{1,n}$ em [2].

As matrizes de ativação são definidas como:

$$f_{1,kl}(g_{kl}) = U(t) = F(A^T P(I+SP)^{-1}A - P + Q)$$
$$f_{2,kl}(h_{kl}) = Y(t) = F(LL^T - P)$$

Obtém-se as seguintes equações dinâmicas neurais para resolver a $EARD$, (5), utilizando uma RNR

$$\begin{array}{rcl} \frac{dV(t)}{dt} = & & -\eta_v[U(t)^T A^T (I+SV(t))^{-1}A - A^T V(t)(I+SV(t))^{-1} \\ & & S^T U(t)(I+SV(t))^{-1}A - U(t) - Y(t)] \\ \frac{dZ(t)}{dt} = & & -\eta_z Z(t) Y^T(t) \\ U(t) = & & F(A^T V(t)(I+SV(t))^{-1}A - V(t) + Q) \\ Y(t) = & & F(Z(t)Z^T(t) - V(t)) \end{array} \tag{10}$$

Detalhes do desenvolvimento completo do processo de análise para a obtenção das equações dinâmicas neurais que descrevem a $EARD$ pode ser achado no artigo [2].

A arquitetura desta nossa RNR, está constituída de quatro camadas conectadas bidirecionalmente. A camada 1 (ou primeira) é de entrada, representada por $U(t) = [u_{ij}(t)]$, a camada 4 (ou última) de saída, representada por $V(t) = [v_{ij}(t)]$ e duas camadas intermédiarias representadas por $Y(t) = [y_{ij}(t)]$ e $Z(t) = [z_{ij}(t)]$, respectivamente. A partir disso, podemos fazer P corresponder à saída final da rede $(v_{ij}(t))$. A matriz de ativação de estado $V(t)$ para os neurônios na camada de saída representa o resultado computacional de P (solução da equação (5)) e a matriz de estado $Z(t)$ representa o fator Cholesky de P, ou seja L. As camadas para $V(t), U(t)$ e $Y(t)$ vão consistir de arranjos quadrados $(n \times n)$ de neurônios.

IV. Equação Dinâmica Neural Contínua

Nesta seção vamos apresentar as equações dinâmicas neurais que descrevem a equação de Riccati contínua utilizando RNR, detalhes do procedimento em [11].

Para motivação considere um sistema linear invariante e contínuo no tempo completamente controlável:

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{11}$$

cuja lei de controle, $u(t) = -Kx(t)$, depende de uma matriz de ganho, K, que vai estar definida através de uma matriz P correspondente à solução da equação matricial algébrica de Riccati Contínua:

$$A^T P + PA - PSP + Q = 0 \tag{12}$$

onde $S = BR^{-1}B^T$ é uma matriz simétrica e semidefinida positiva.

Para fazer a formulação utilizando uma RNR na solução da equação (12), retomaremos apenas as equações que foram definidas para o caso discreto, no seção anterior.

Seja a função objetivo definida por:

$$\begin{array}{rcl} \min E(P,L) & = & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}[G(P)] + e_{ij}[H(P,L)] \\ & onde & \\ G(P) & = & PSP - A^T P - PA - Q \\ H(P,L) & = & LL^T - P \end{array} \tag{13}$$

As funções de ativação são definidas como:

$$U(t) = F(V(t)SV(t) - A^T V(t) - V(t)A - Q)$$
$$Y(t) = F(Z(t)Z^T(t) - V(t))$$

As equações dinâmicas neurais que descrevem a $CARE$ utilizando uma RNR, são apresentadas a seguir:

$$\begin{array}{ll} \frac{dV(t)}{dt} & = -\eta_p[V(t)SU(t) + U(t)SV(t) - AU(t) - U(t)A^T - Y(t)] \\ \frac{dZ(t)}{dt} & = -\eta_l Y(t)Z(t) \\ U(t) & = F(V(t)SV(t) - A^T V(t) - V(t)A - Q) \\ Y(t) & = F(Z(t)Z^T(t) - V(t)) \end{array} \tag{14}$$

V. Implementações e Resultados

Nesta seção, discutimos os resultados das implementações neurais das equações algébricas de Riccati discreta ($EARD$) e contínua ($EARC$) no tempo com vários exemplos. As implementações para resolver a **EARD Neural** e a **EARC Neural** em forma geral, foram desenvolvidas no Matlab usando o método de Runge-Kutta de quarta ordem; em nosso caso as equações diferenciais coincidem com as equações dinâmicas neurais obtidas em cada problema analisado: Discreto e Contínuo, respectivamente. A estrutura do algoritmo para calcular $V(t)$, por exemplo, no caso discreto é:

s_1 = feval('fv',etav,V,S,U,A);
s_2 = feval('fv',etav,V+h*s_1/2,S,U,A);
s_3 = feval('fv',etav,V+h*s_2/2,S,U,A);
s_4 = feval('fv',etav,V+h*s_3,S,U,A);
V = V + h*(s_1 + 2*s_2 + 2*s_3 + s_4)/6;
Vout = [Vout;V]

Cada s_i corresponde a uma variável que é encarregada da atualização de um peso sináptico da rede ou equivalentemente à variável desconhecida do problema, x.

Exemplo 1. Considere o controle RLQ do seguinte sistema contínuo no tempo instável em malha aberta, onde os coeficientes das matrices são:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1.6 & 0.9 \\ -0.1 & 2.1 \end{bmatrix}$$
$$Q = \begin{bmatrix} 1.5 & -1 \\ -1 & 5 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{15}$$

Resolvemos a equação neural de Riccati Contínua com a mesma arquitetura da RNR e com a implementação do algoritmo numérico da RNR explicado detalhadamente em [11]. A solução da respectiva equação é:

$$P = \begin{bmatrix} 1.222 & -0.671 \\ -0.671 & 1.339 \end{bmatrix}$$

A simulação foi realizada usando os seguintes parâmetros como condição inicial, $(\eta_p, \eta_l) = (10, 1000)$, $\Delta t = 10^{-5}$, $P(0) = \begin{bmatrix} 10 & 1 \\ 1 & 10 \end{bmatrix}$ e $Y(0) = I$.

A Figura 1 apresenta as trajetórias de $Z(t)$ e $V(t)$, respectivamente. A partir destas figuras podemos observar que tanto $Z(t)$ como $V(t)$ atingem seus valores estáveis rapidamente.

Neste exemplo estamos simplesmente verificando e calibrando nossa implementação computacional da $EARC$ neural para futuras utilizações em outras equações, por exemplo as equações de Lyapunov. Temos como referência o artigo [11], onde a simulação, para o mesmo exemplo, foi feita em código C. Neste caso fizemos uma comparação entre os resultados obtidos no trabalho de referência e os obtidos em nossa implementação neural e chegamos a conclusão que a nossa proposta neural também está correta.

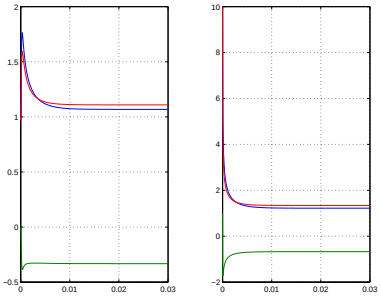

Fig. 1. Resultados da equação neural de Riccati Contínua

Portanto, nossa proposta e implementação para resolver a **EARC** utilizando RNR, ($EARC\ Neural$), apresenta bons resultados que podem ser testados e comprovados com qualquer outro método de solução.

Exemplo 2. Este exemplo considera o problema de controle ótimo linear quadrático para o caso discreto. O desempenho da solução da **EARD**, usando uma abordagem com rede neural recorrente multicamada é apresentado. Os parâmetros para as matrizes do sistema são:

$$A = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \quad B = \begin{bmatrix} 4.877 & 4.877 \\ -1.1895 & 3.569 \end{bmatrix}$$
$$Q = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.02 \end{bmatrix} \qquad R = \begin{bmatrix} 0.33 & 0 \\ 0 & 3 \end{bmatrix}$$

As equações dinâmicas neurais (10) foram utilizadas para resolver a equação de Riccati Discreta (5). A solução P, K obtida para a respectiva **EARD Neural** é:

$$P = \begin{bmatrix} 0.0104 & 0.0032 \\ 0.0032 & 0.0504 \end{bmatrix} \qquad K = \begin{bmatrix} 0.0715 & -0.0705 \\ 0.0136 & 0.0455 \end{bmatrix}$$

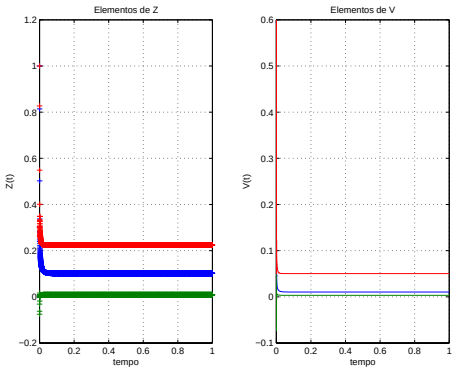

Fig. 2. Resultados de V(t) e Z(t) para a EARD usando RNR

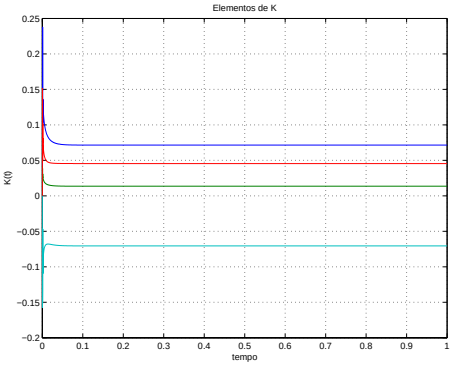

Fig. 3. Resultados de K(t), para a EARD usando *RNR*

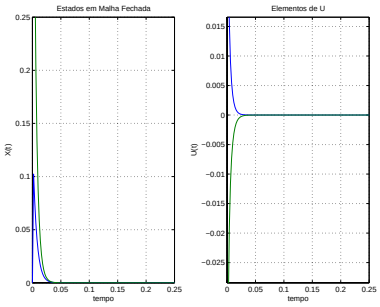

Fig. 4. Resultados de x(t) e u(t), usando *RNR*

Esta matriz P é simétrica, definida positiva e satisfaz (5). A simulação foi realizada usando as equações dinâmicas neurais implementadas, onde $(\eta_p, \eta_l) = (3000, 10000)$, $\Delta t = 10^{-5}$ $P(0) = [0.1]_{2\times 2}$ e $Y(0) = I$. As Figs.2-4 apresentam as trajetórias de $Z(t), V(t), K(t), x(t), u(t)$ para a solução da equação **EARD Neural** implementada neste exemplo . A partir da Fig. 2, podemos observar que os elementos de $V(t)$ atingem seu valor estável rapidamente. Esta solução neural corresponde à solução dada por Laub em [3] cujo método é uma variante da abordagem de autovetor clássica usando um conjunto de vetores Schur.

Portanto, nossa proposta e implementação para resolver a **EARD** utilizando *RNR*, apresenta bons resultados que podem ser testados e comprovados com qualquer outro método de solução, em especial [1] [6].

Exemplo 3. Considere o problema de controle RLQ de um sistema discreto com parâmetros variantes no tempo definidos na matrix B:

$$A = \begin{bmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{bmatrix} \quad Q = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.02 \end{bmatrix}$$

$$B = \begin{bmatrix} 4.877 + 2sin((2\pi t)/5) & 4.877 \\ -1.1895 & 3.569 \end{bmatrix} \quad R = \begin{bmatrix} 0.33 & 0 \\ 0 & 3 \end{bmatrix}$$

$$x_0 = \begin{bmatrix} 0.9516 \\ 0.2603 \end{bmatrix}$$

Sejam as matrices iniciais $V = 0$ e $Z = I$, x_0 é um vetor de estado inicial aleatório.

A solução P, K obtida para a respectiva **EARD Neural** é:

$$P = \begin{bmatrix} 0.0098 & 0.0032 \\ 0.0032 & 0.0506 \end{bmatrix} \quad K = \begin{bmatrix} 0.0687 & -0.0715 \\ 0.0131 & 0.0455 \end{bmatrix}$$

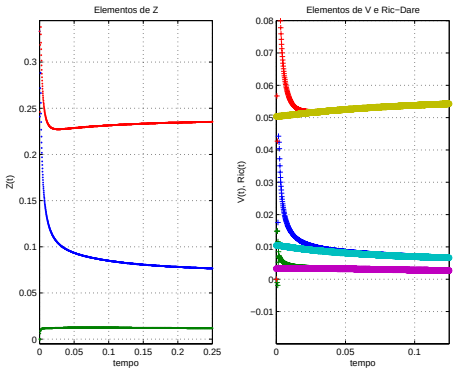

Fig. 5. Resultados de Z(t) e V(t) usando *RNR*

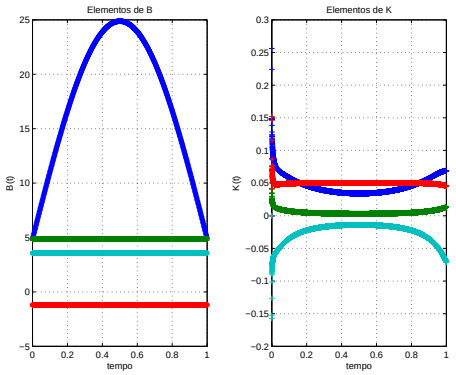

Fig. 6. Resultados de B(t) e K(t) usando *RNR*

Simulações foram realizadas usando as equações dinâmicas neurais implementadas, com $(\eta_p, \eta_l) = (3000, 10000)$. Como $B(t)$ é uma matriz variante no tempo, resultados de simulações em tempo real são apresentadas nas Figs.5-7, ilustrando os resultados de $Z(t)$, $V(t)$, $B(t)$, $K(t)$, $x(t)$ e $u(t)$. Pode-se observar que apesar dos valores da matrix $B(t)$ mudarem, o resto das matrices, $Z(t)$, $V(t)$ e $K(t)$ acompanham as mudanças de forma a satisfazer(5).

Na segunda figura em Fig.5 apresentamos uma comparação entre a solução obtida para a equação **EARD Neural** a cada $0.05s$ com a solução exata da equação algébrica de Riccati discreta para este caso analisado. A partir desta mesma Figura, podemos observar como os elementos de ambas matrices $Z(t)$ e $V(t)$ atingem seus valores estáveis rapidamente.

A cada $0.05s$ temos um modelo novo da planta. Este modelo é obtido, deixando a matriz $B(t)$ fixa durante este

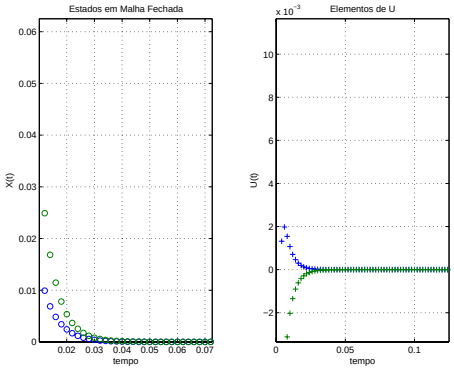

Fig. 7. Resultados de x(t) e u(t) com *RNR*

intervalo de tempo e fazendo 10 iterações para resolver a *EARD Neural*. Logo, a cada $0.05s$ obtem-se uma nova solução *EARD Neural* em $V(t)$, calcula-se então o ganho K e aplica-se a lei de controle realimentado em malha fechada na planta. A cada $0.05s$ temos um novo estado da planta.

A Fig.7 apresenta respectivamente, o vetor de estado $x(t)$ e a trajetória do vetor de controle $u(t)$ no sistema de controle realimentado. Vale ressaltar que o vetor $u(0)$ é não nulo, pois ele é obtido apartir de uma solução da *EARD Neural* $V(t)$ iterada durante $0.05s$.

VI. Conclusiones

Os algoritmos para solução neural da *EARD* e da *EARC* aqui implementados e ilustrados nos exemplos de aplicação a problemas de controle RLQ tendo em mente aplicações em tempo real, mostram-se viáveis, eficientes e úteis, além de confirmarem a exatidão de cada solução obtida por métodos diversos, em especial pelo que propuzemos em [2].

Agradecimentos

Os autores agradecem a FAPESP; assim como os comentários e contribuições do Prof. Dr. João Vianna da Fonseca Neto, UFMa, Mauricio Bordon e Sergio Antonio Augusto Filho, UNICAMP.

References

[1] Tamariz ADR. Uma proposta metodologica para solução da equação algébrica de riccati em formas sequencial e paralela. Master's thesis, Universidade Estadual de Campinas, UNICAMP, Abril 1999.

[2] Tamariz ADR. and Bottura CP. Discrete-time systems neuro-riccati equation solution. *IEEE International Joint Conference on Neural Networks, IJCNN-2005*, August 2005.

[3] Alan J. Laub. A Schur Method for Solving Algebraic Riccati Equations. *IEEE Transactions on Automatic Control*, AC-24(6), December 1979.

[4] W.F. Arnold and A.J. Laub. *Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations. Proceedings of the* IEEE, 72(12), 1984.

[5] Bottura CP., Tamariz ADR., Fonseca JV., and Gilmar Barreto. Sequential and parallel algebraic riccati equations solutions via esst on the schur method. *Proceedings of the* 38^{th} *Conference on Decision and Control, CDC*, pages 2739–2740, 1999.

[6] Wang J. Recurrent neural networks for solving linear matrix equations. *Computers & Mathematical with applications*, 26(9):23–34, 1993.

[7] Wang J. Solving simultaneous linear equations using recurrent neural networks. *Information Sciences*, 76(3-4):255 – 277, 1994.

[8] Chun-Liang Lin, Chi-Chih Lai, and Teng-Hsien Huang. A neural network for linear matrix inequality problems. *IEEE Transactions on Neural Networks*, 11(5), September 2000.

[9] Haykin Symon. *Neural Networks: A Comprehensive Foundation*. Second Edition, Prentice Hall, 1999.

[10] D.R. Vaughan. A Nonrecursive Algebraic Solution for the Discrete Riccati Equation. *IEEE Transactions on Automatic Control*, AC-15(5), October 1970.

[11] Jun Wang and Guang Wu. A multilayer recurrent neural network for solving continuos-time algebraic riccati equations. *Neural Networks*, 11:939–950, 1998.

[12] Doyle JC Zhu K and Glover K. *Robust and Optimal Control*. Prentice Hall, 1996.